Leisure-Management-Web-App

Jakob Pirker, 1231394

10. Juni 2016

Inhaltsverzeichnis

1 Motivation und Ziele

In fast allen Vereinen und Gemeinschaften werden öfters Freizeiten angeboten. Damit sind Veranstaltungen gemeint, bei denen Mitglieder gemeinsam (meistens für mehrere Tage) an einen bestimmten Ort fahren, um dort gemeinsam Zeit zu verbringen, um einer bestimmten Tätigkeit intensiv nachzukommen (z.B. Trainingslager) oder Ähnliches. Mit der Anzahl der Teilnehmer steigt auch der Organisationsaufwand, und erreicht oft ein Maß bei dem die Organisation durch herkömmliche Methoden wie Absprache und Papier und Stiftünwirtschaftlich bis unmöglich wird.

Ziel dieser Arbeit war es, eine Basis zu schaffen, die die Organisation einer solchen Freizeit vereinfacht. Dies soll dadurch geschehen, dass die Anwendung eine Struktur für die Organisation vorgibt, in der oft benötigte Elemente bereits integriert sind, und direkt für die Organisation verwendet werden können. Einige Beispiele hierfür wären: Auflistung aller Teilnehmer, Überprüfung der Anwesenheiten und Veranstaltungsbeiträge jedes Teilnehmers, Aufgabenverteilungen... Die grundlegenden Anforderungen sind in den nächsten Punkten aufgelistet.

1.1 Webservice

Eine wichtige Anforderung die sich aus der Verteilung von Aufgaben vergibt ist die Bedienbarkeit von einer beliebigen Stelle aus. Aus dieser Anforderung heraus hat sich die Ausführung der Anwendung als Web-Applikation ergeben. Daraus ergeben sich folgende Vorteile:

- Bedienbarkeit von jedem internetfähigen Endgerät mit Browser
- keine Notwendigkeit spezieller Software
- zentrale Datenverwaltung

1.2 Anpassbarkeit und Erweiterbarkeit

Die vorgegebene Struktur soll kein absoluter Maßstab sein, sondern der Anwender soll (wo möglich) selbst entscheiden können, welche Elemente er verwendet, und welche nicht. Außerdem soll die Applikation von einer anderen Person (mit Infromatik-Hintergrundwissen) möglichst einfach gewartet und an eine spezielle Freizeit angepasst werden können. Dies bezieht sich sowohl auf das Hinzufügen und Entfernen von Content auf der Client-Seite, als auch auf das Hinzufügen und Entfernen von zusätzlichen bzw. unnötigen Informations-Attributen und Funktionen auf der Server-Seite. Außerdem soll es möglich sein größere Informationselemente (z.B. Events) ohne viel Aufwand in die Datenstruktur zu integrieren.

1.3 Anforderungsübersicht

Es soll möglich sein Personen einzutragen, die Teilnehmer und/oder Mitarbeiter sein können, und für Personen Adressen festzulegen. Teilnehmer können einer

Unterkunft zugewiesen werden, und für diese kann wiederum eine Adresse festgelegt werden. Jedem Mitarbeiter können mehrere Aufgaben zugewiesen werden. Jeder Person können mehrere Zahlungen zugewiesen werden, eine Zahlung kann wiederum einem Zahlungsdepot zugewiesen werden.

2 Problemstellungen und Lösungsansätze

2.1 REST

Representational State Transfer (abgekürzt REST, seltener auch ReST) bezeichnet ein Programmierparadigma für verteilte Systeme, insbesondere für Webservices.[1]

REST verwiest auf einige Prinzipien, die vorausgesetzt werden, damit der implementierte Service als RESTful bezeichnet werden kann. Bei den Prinzipien handelt es sich um: Client-Server, Zustandslosigkeit, Caching, Einheitliche Schnittstelle, Mehrschichtige Systeme und Code on Demand (optional). Im Zuge des Projekts wurde Wert darauf gelegt, dass das System diese Eigenschaften erfüllt.

2.2 Überblick

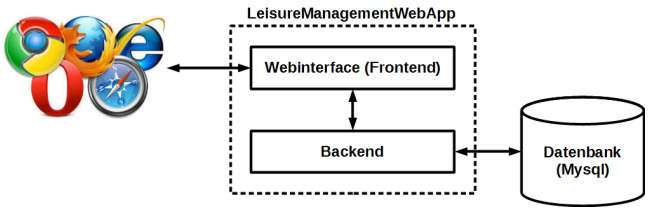

Abbildung 1: Grober Design Überblick

Wie die meisten Web-Applikationen besteht das Projekt aus einem Front- und einem Backend. Das Frontend (Client-Side) besteht aus einer Homepage, die von einem Browser aus aufgerufen wird. Es stellt eine Oberfläche zur Verfügung, mit deren Hilfe der Benutzer auf die Funktionalitäten der Applikation zugreifen kann. Das Frontend enthält jedoch nur Anzeige- und Transaktionslogik. Das bedeutet, dass es zwar Zugriff auf Funktionalitäten verschafft, und die Anzeigeelemente entsprechend der Benutzerinteraktion verändert, dann aber nur mittels eines http-Requests die entsprechenden Daten vom Backend anfordert und den zugehörigen Anzeigeelementen zuordnet. Die eigentliche Logik liegt im Backend (Server-Side). Es empfängt die Requests des Frontends, und sendet Daten als Antwort auf die Requests. So gut wie alle für die Organisation der Freizeit

[1] `https://de.wikipedia.org/wiki/Representational_State_Transfer(10.6.16)`

notwendigen Daten werden in einer Datenbank abgelegt. Das Backend fungiert auch als Schnittstelle zur Datenbank.

2.3 Backend: Spring MVC

Das Backend wurde in Java implementiert. Hier wurde Spring MVC als unterstützendes Framework gewählt, da es einen sehr großen Funktionsumfang hat. Die beiden für dieses Projekt wichtigsten Features werden im Folgenden kurz beschrieben.

2.3.1 Datenbank Anbindung

Spring stellt mittels Spring-Data einige sehr praktische Werkzeuge zur Verfügung mit denen auf relationale Datenbanken zugegriffen werden kann. Es werden JPA-Schnittstellen zur Verfügung gestellt, die die Zuordnung und Übertragung von Objekten zu Datenbankeinträgen sehr stark vereinfachen.

2.3.2 Datenübertragung

Spring unterstützt außerdem einfache Wege, mittels denen URL's einfach auf Methoden gemappt, und die empfangenen http-Requests durch Lesen der Parameter in der URL oder der Daten im http-Body in Nutzbare Daten umgewandelt werden können.

2.4 Frontend: AngularJS/Javascript

Zur Erstellung des Frontends wurde das JavaScript Framework AngularJS verwendet. AngularJS erweitert HTML um Attribute. Diese Attribute schaffen erstens eine sehr gute Schnittstelle zwischen den HTML-Elementen und der dahinter liegenden JavaScript Logik und zweitens ermöglichen sie eine dynamische Anpassung der HTML-Struktur des Basisdokuments.

3 Backend

3.1 Spring

3.1.1 Dependency Injection

3.2 Architektur

3.2.1 Controller

3.2.2 Services

3.2.3 Repositories

3.3 Datenbankanbindung

3.4 Frontend-Interface

4 Frontend

4.1 Angular JS

4.1.1 Zwei-Weg Datenbindung

4.2 Tabs

4.3 Tabellen

4.4 Eingabeformen

5 Installation und Anmerkungen

6 Quellen

- REST: https://de.wikipedia.org/wiki/Representational_State_Transfer
- REST: https://spring.io/understanding/REST
- JPA: https://de.wikipedia.org/wiki/Java_Persistence_API

6.1 AngularJS

- https://de.wikipedia.org/wiki/AngularJS
- https://angularjs.org/
- http://www.w3schools.com/angular/

6.2 Abbildungen

- Abbildung 1 (Browser Zusammenstellung): http://vestavialibrary.org/wp-content/uploads/2016/02/web-browsers.jpg